

XAML

Extensible Application Markup Language

Manuel Naujoks (IB3)

Agenda

- Einführung
 - Beschreibungssprachen
 - Vorteile
 - Nachteile
- XAML technisch
 - Aufbau
 - Kompilierung
 - Verarbeitung zur Laufzeit
 - Routed Events
- Zusammenfassung

Beschreibungssprachen

Vorteile

Nachteile

EINFÜHRUNG

Beschreibungssprachen für Oberflächen

- XUL (XML User Interface Language)
- MXML (MX Markup Language)
- LZX (Laszlo XML)
- XHTML (Extensible Hypertext Markup Language)
- ZUML (ZK User Interface Markup Language)
- SwingML (Swing Markup Language)
- ...

● XAML

Beschreibungssprache XAML

- Extensible Application Markup Language
- XML Syntax
- Instanziierung von Objekten
- .NET 3.0 (Windows Presentation Foundation)
 - Deklarative Oberflächenbeschreibung
- Austauschformat für Entwickler und Designer
 - Trennung von Logik und Aussehen des UI

Vorteile

- Kompilierung, Laden zur Laufzeit
- Verwendung (WPF, WF, Silverlight)
- Sehr gute Tool-Unterstützung
 - Microsoft Visual Studio
 - Microsoft Expression Studio
- Alle Vorteile von .NET
 - Sprach-Unabhängigkeit
 - Umfang von .NET



Nachteile

- Geht nur für .NET
- Keine Plattformunabhängigkeit
- Keine Cascading Style Sheets

Lösungen

- Mono (OpenSource .NET Framework)
- eFace (XAML & WPF für Java)



Aufbau

Kompilierung

Verarbeitung zur Laufzeit

Routed Events

XAML TECHNISCH

Aufbau „Window1.xaml“

```
<Window x:Class="WpfApplication1.Window1"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="Window1" Height="300" Width="300">
    <Grid>
    </Grid>
</Window>
```

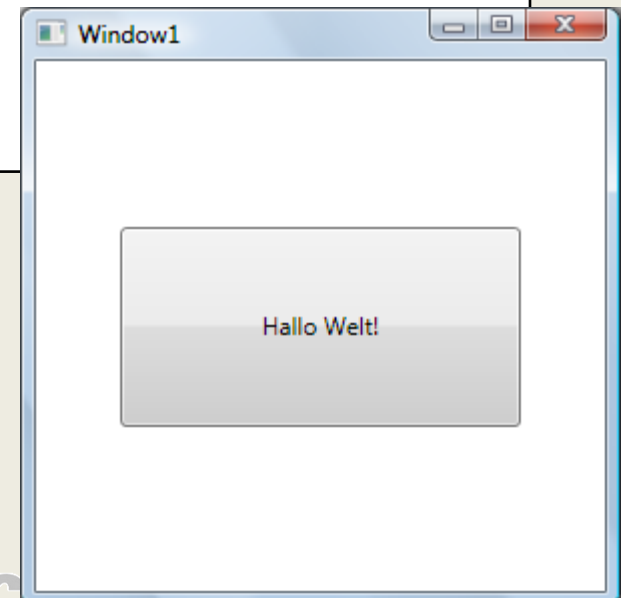
- <http://schemas.microsoft.com/winfx/2006/xaml/presentation> (WPF Elemente)
- <http://schemas.microsoft.com/winfx/2006/xaml> (System.Windows.Markup)

Eigene Namespaces

clr-namespace:System;assembly=mscorlib

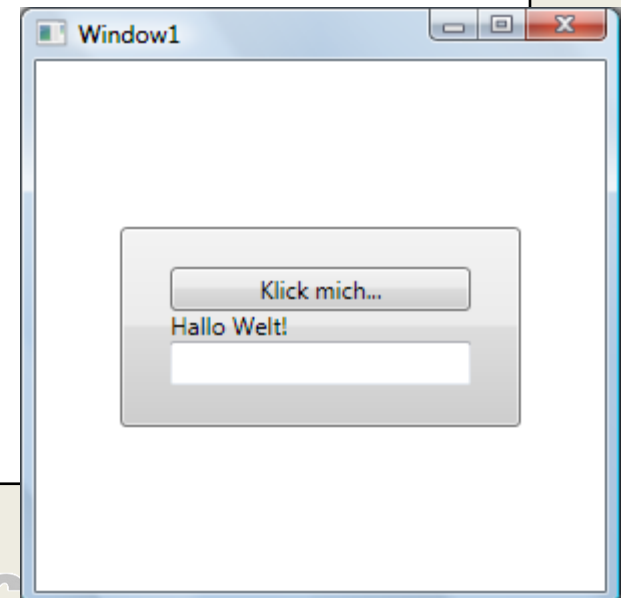
Aufbau „Window1.xaml“ Beispiel

```
<Window x:Class="WpfApplication1.Window1"
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  Title="Window1" Height="300" Width="300">
  <Grid>
    <Button Height="100" Width="200" Name="myButton">
      Hallo Welt!
    </Button>
  </Grid>
</Window>
```

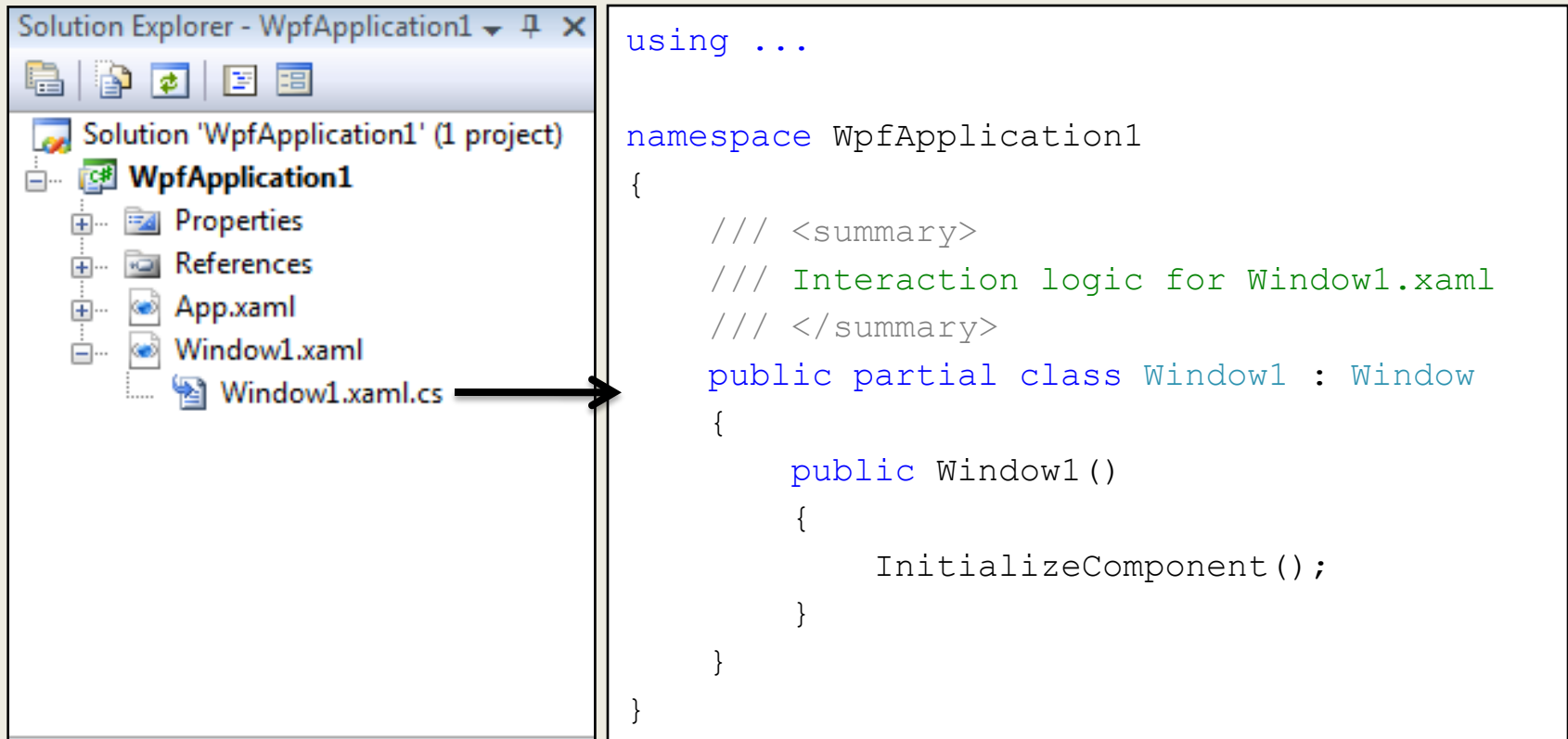


Aufbau „Window1.xaml“ Beispiel

```
<Window x:Class="WpfApplication1.Window1"
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  Title="Window1" Height="300" Width="300">
  <Grid>
    <Button Height="100" Width="200" Name="myButton">
      <StackPanel Width="150">
        <Button Content="Klick mich..."/>
        <TextBlock>
          Hallo Welt!
        </TextBlock>
        <TextBox/>
      </StackPanel>
    </Button>
  </Grid>
</Window>
```



Aufbau „Window1.xaml.cs“



The image shows a screenshot of a Visual Studio Solution Explorer on the left and the corresponding code behind file on the right. The Solution Explorer displays a project named 'WpfApplication1' with files 'App.xaml', 'Window1.xaml', and 'Window1.xaml.cs'. An arrow points from 'Window1.xaml.cs' to the code editor on the right. The code editor shows the following C# code:

```
using ...

namespace WpfApplication1
{
    /// <summary>
    /// Interaction logic for Window1.xaml
    /// </summary>
    public partial class Window1 : Window
    {
        public Window1()
        {
            InitializeComponent();
        }
    }
}
```

- Code Behind File (C#)

Aufbau „Window1.g.cs“

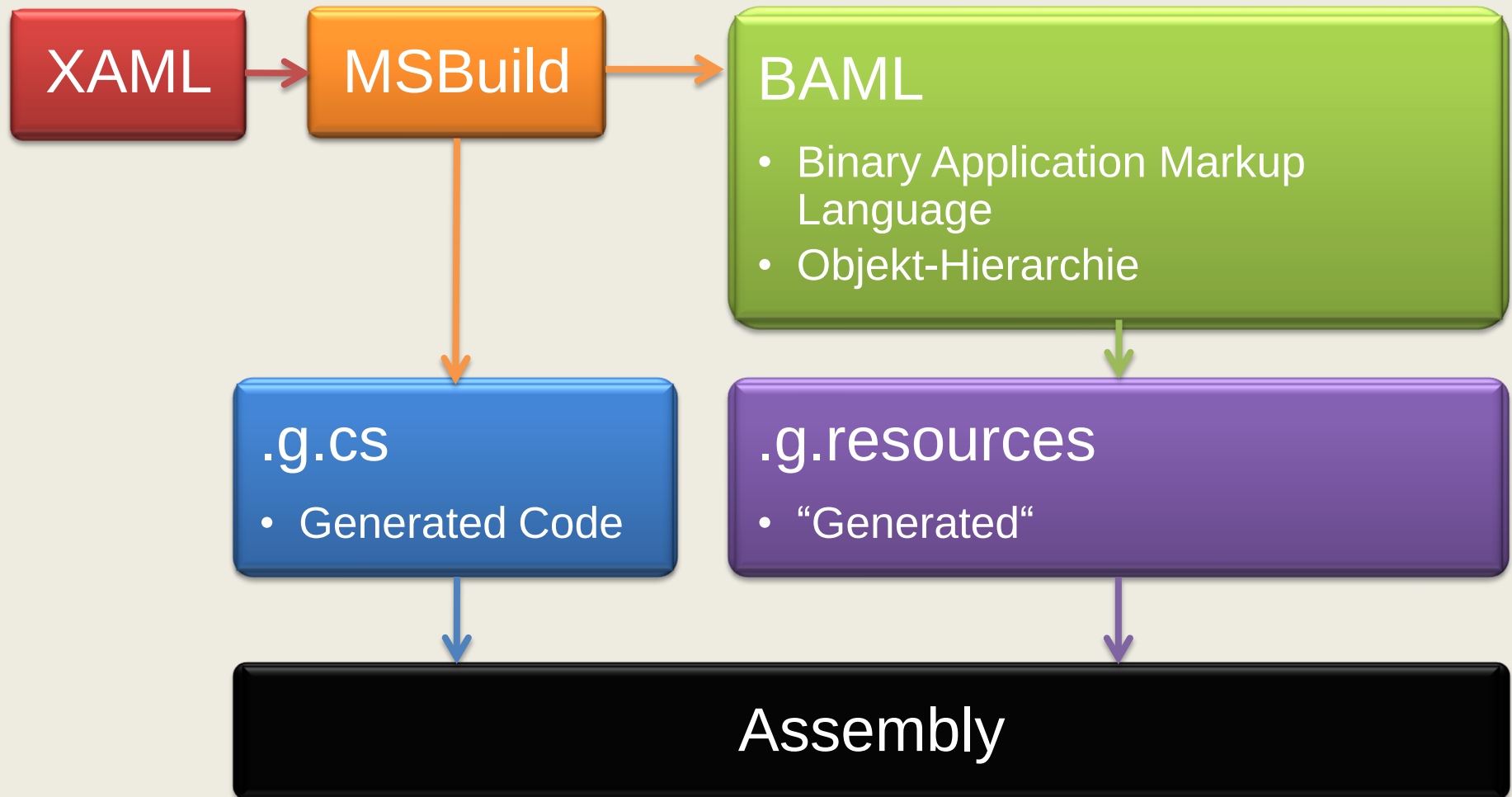
```
#pragma checksum "..\..\Window1.xaml" "{406ea660-64cf-4c82-b6f0-42d48172a799}"  
"AB4CDA123E53A53945B84FC6C78C2C0E"  
  
//-----  
// <auto-generated>  
//     This code was generated by a tool.  
//     Runtime Version:2.0.50727.1378  
//  
//     Changes to this file may cause incorrect behavior and will be lost if  
//     the code is regenerated.
```

```
public partial class Window1: System.Windows.Window
```

```
internal System.Windows.Controls.Button myButton;
```

- Generated Code File (C#)

Kompilierung



Verarbeitung zur Laufzeit

- Laden von XAML oder BAML Inhalt über URI
 - Application.LoadComponent (Uri) : Object
 - Application.LoadComponent (Object, Uri)
- Laden von XAML Inhalt aus Stream
 - XamlReader.Load (Stream) : Object
- Zugriff auf Elemente der Zielkomponente?

Verarbeitung zur Laufzeit

```
public partial class Window1 : System.Windows.Window,
    System.Windows.Markup.IComponentConnector

public void InitializeComponent() {
    if (_contentLoaded) {
        return;
    }
    _contentLoaded = true;
    System.Uri resourceLocator =
        new System.Uri("/WpfApplication1;component/window1.xaml",
            System.UriKind.Relative);

    System.Windows.Application.LoadComponent(this, resourceLocator);
}

void System.Windows.Markup.IComponentConnector.Connect(
    int connectionId, object target) {
    switch (connectionId) {
        case 1:
            this.myButton = ((System.Windows.Controls.Button) (target));
            return;
    }
    this._contentLoaded = true;
}
```

Generated Code File

Routed Events

- Direkter Event
 - `Click`
- Bubbling Event
 - `MouseLeftButtonDown`
- Tunneling Event
 - `PreviewMouseLeftButtonDown`
 - Mit Prefix “Preview“

Nicht jedes Event hat ein Bubbling oder Tunneling Event!
MSDN Library!

EventHandler

Beispiel

```
<Button
PreviewMouseLeftButtonDown="Button_PreviewMouseLeftButtonDown_1"/>
<x:Code>
    <![CDATA[
private void Button_PreviewMouseLeftButtonDown_1(object sender,
MouseButtonEventArgs e)
{
    MessageBox.Show("Hallo");
    e.Handled = true;
}
]]>
</x:Code>
```

- Inline EventHandler
- Code Behind EventHandler

Zuweisung von EventHandlern

```
void System.Windows.Markup.IComponentConnector.Connect(  
    int connectionId, object target) {  
    switch (connectionId)  
    {  
        case 1:  
            this.myButton = ((System.Windows.Controls.Button) (target));  
            this.myButton.Click +=  
                new System.Windows.RoutedEventHandler(this.Button_Click);  
            return;  
        }  
        this._contentLoaded = true;  
    }  
}
```

- Generated Code Datei
- Zugriff auf XAML Element auch ohne Namen

Routed Events

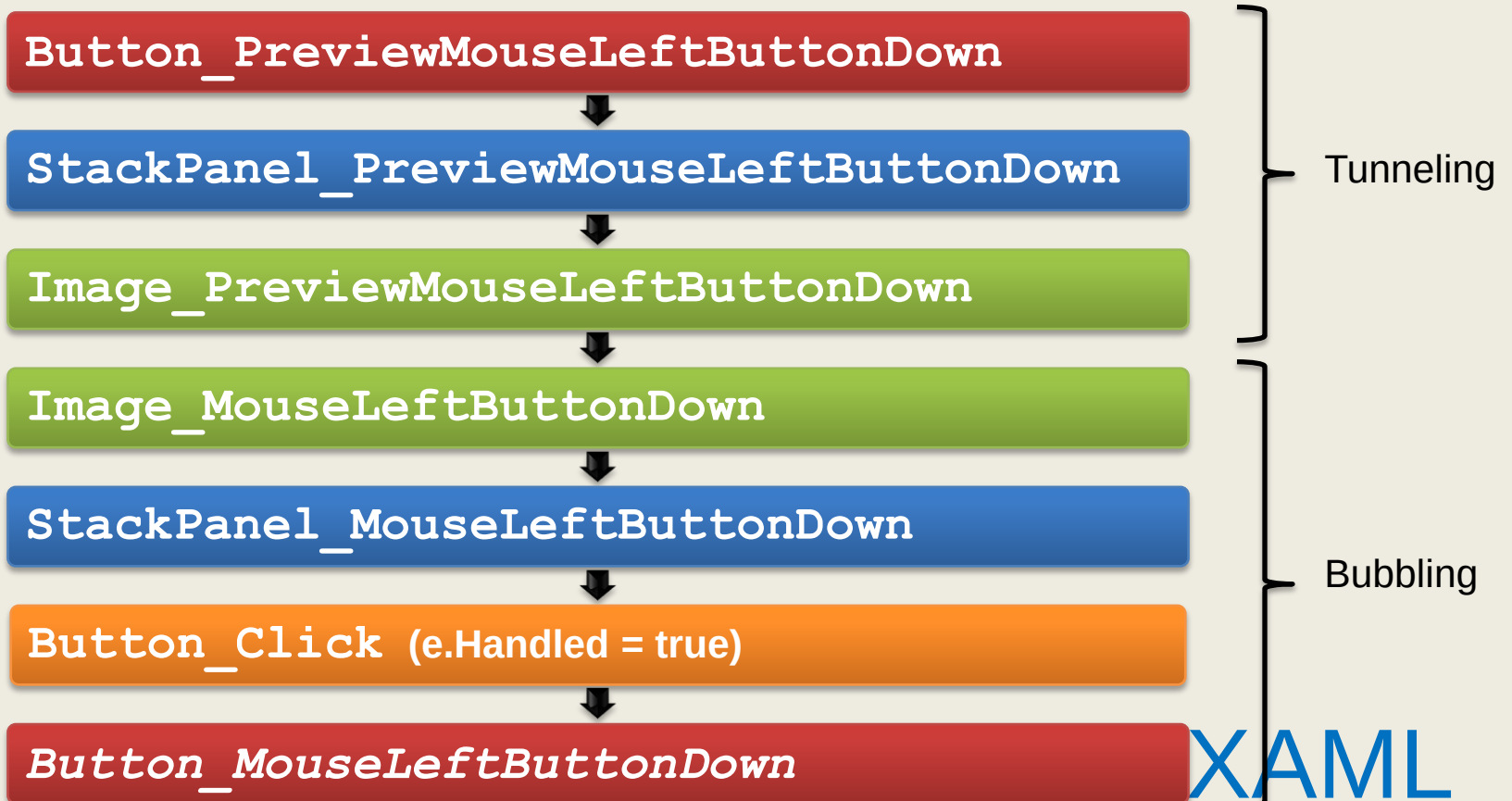
Beispiel

```
<Button
  MouseLeftButtonDown="Button_MouseLeftButtonDown"
  PreviewMouseLeftButtonDown="Button_PreviewMouseLeftButtonDown"
  Click="Button_Click">
  <StackPanel
    MouseLeftButtonDown="StackPanel_MouseLeftButtonDown"
    PreviewMouseLeftButtonDown="StackPanel_PreviewMouseLeftButtonDown">
    <Image Width="130" Source="D:\temp\billg1.jpg"
    MouseLeftButtonDown="Image_MouseLeftButtonDown"
    PreviewMouseLeftButtonDown="Image_PreviewMouseLeftButtonDown">
    <TextBlock FontSize="15">Der große Bill!</TextBlock>
  </StackPanel>
</Button>
```



Routed Events

- Mausklick auf das Bild



ZUSAMMENFASSUNG

Zusammenfassung

- Kompilierung
 - BAML, Code Behind Class, Verarbeitung zur Laufzeit
- .NET 3.0
 - WPF, Silverlight, Mono, eFace
- Beliebige Objekt-Hierarchien
 - `xmlns:my="clr-namespace:MyNamespace;assembly=WpfApp"`
- Bubbeling, Tunneling, Direct Events

Bessere Zusammenarbeit von Designern und Entwicklern.

Weitere Informationen

- XAML

<http://msdn2.microsoft.com/en-us/library/ms747122.aspx>

- WPF

<http://msdn2.microsoft.com/en-us/library/ms754130.aspx>

- eFace

<http://www.soyatec.com/eface/>

- Silverlight

<http://msdn2.microsoft.com/en-us/library/bb404300.aspx>

- nama0017@hs-karlsruhe.de

- hallo-welt.spaces.live.com